```
RRRRRRRRRRR      MMM          MMM     SSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSSSSS
RRR         RRR  MMMMMM    MMMMMM   SSS
RRR         RRR  MMMMMM    MMMMMM   SSS
RRR         RRR  MMMMMM    MMMMMM   SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRR         RRR  MMM  MMM  MMM  MMM  SSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRRRRRRRRRRR     MMM          MMM     SSSSSSSSS
RRR   RRR        MMM          MMM             SSS
RRR   RRR        MMM          MMM             SSS
RRR   RRR        MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR      RRR     MMM          MMM             SSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
RRR         RRR  MMM          MMM  SSSSSSSSSSS
```

FILEID**RM1INPSCN

```
RRRRRRRR    MM      MM      11      IIIIII   NN      NN  PPPPPPPP     SSSSSSSS    CCCCCCCC  NN      NN
RRRRRRRR    MM      MM      11      IIIIII   NN      NN  PPPPPPPP     SSSSSSSS    CCCCCCCC  NN      NN
RR      RR  MMMM  MMMM    1111        II     NN      NN  PP      PP   SS          CC        NN      NN
RR      RR  MMMM  MMMM    1111        II     NN      NN  PP      PP   SS          CC        NN      NN
RR      RR  MM  MM  MM      11        II     NNNN    NN  PP      PP   SS          CC        NNNN    NN
RR      RR  MM  MM  MM      11        II     NNNN    NN  PP      PP   SS          CC        NNNN    NN
RRRRRRRR    MM      MM      11        II     NN  NN  NN  PPPPPPPP     SSSSSS      CC        NN  NN  NN
RRRRRRRR    MM      MM      11        II     NN  NN  NN  PPPPPPPP       SSSSSS    CC        NN  NN  NN
RR  RR      MM      MM      11        II     NN    NNNN  PP                   SS  CC        NN    NNNN
RR  RR      MM      MM      11        II     NN    NNNN  PP                   SS  CC        NN    NNNN
RR    RR    MM      MM      11        II     NN      NN  PP                   SS  CC        NN      NN      ....
RR    RR    MM      MM      11        II     NN      NN  PP                   SS  CC        NN      NN      ....
RR      RR  MM      MM    111111    IIIIII   NN      NN  PP           SSSSSSSS      CCCCCCCC  NN      NN      ....
RR      RR  MM      MM    111111    IIIIII   NN      NN  PP           SSSSSSSS      CCCCCCCC  NN      NN      ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          $BEGIN  RM1INPSCN,000,RM$RMS1,<SYS$INPUT $, $EOD, & $DECK ROUTINES>
0000     2
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26 ;
0000    27 ;++
0000    28 ; Facility: rms32
0000    29 ;
0000    30 ; Abstract:
0000    31 ;                   this module performs end of file checking, as well as
0000    32 ;                   $deck processing for $get /$find on sys$input
0000    33 ;                   processing for the sequential file organization.
0000    34 ;
0000    35 ;
0000    36 ; Environment:
0000    37 ;                   star processor running starlet exec.
0000    38 ;
0000    39 ; Author:          L F Laverdure,   creation date: 29-JAN-1978
0000    40 ;
0000    41 ; Modified By:
0000    42 ;
0000    43 ;         V03-002 KBT0140         Keith B. Thompson       20-Aug-1982
0000    44 ;                 Reorganize psects
0000    45 ;
0000    46 ;         V03-001 KBT0085         Keith B. Thompson       13-Jul-1982
0000    47 ;                 Clean up some of the psect nonsence
0000    48 ;
0000    49 ;         V02-006 RAS0017         Ron Schaefer    21-Jul-1981
0000    50 ;                 Change buffer management so it can work with stream files.
0000    51 ;                 In particular, this routine only scans the current buffer,
0000    52 ;                 the caller must provide an adequate buffer size.
0000    53 ;
0000    54 ;         V005    REFORMAT        Ken Henderson   30-JUL-1980     6:24
0000    55 ;                 the code was reformatted
0000    56 ;
0000    57 ; Revision History:
```

```
0000     58 ;
0000     59 ;
0000     60 ; 01    L F Laverdure,   17-JUN-1978  20:57
0000     61 ;--     -
0000     62 ;
0000     63
```

```
                        .SBTTL  DECLARATIONS

                ;
                ; Include Files:
                ;

                ;
                ; Macros:
                ;

                        $IRBDEF
                        $IFBDEF
                        $DEVDEF
                        $BDBDEF
                        $PIODEF
                        $RMSDEF

                ;
                ;  macro to call dcl_scan subroutine, setting up in-line argument list
                ;  of match string and equality branch offset
                ;

                        .MACRO  DCL_SCAN STR=,EQUAL=,DISP=W,?L
                        BSB'DISP        DCL_SCAN
                        .ASCIC  %STR%
                L:      .BYTE   EQUAL-L                 ; offset to "equals" routine
                        .ENDM   DCL_SCAN

                ;
                ; Equated Symbols:
                ;

00000009        TAB     = 9                     ; horizontal tab
00000061        LOWERCASE_A     = 97            ; lower case a
0000007A        LOWERCASE_Z     = 122           ; lower case z

0000000F        EODSTR_MAXLEN   = 15            ; max. len. of end-of-data scan string

                ;
                ;
                ; Own Storage:
                ;

```

```
0000   109          .SBTTL  RM$INPUT_SCAN - CHECK SYS$INPUT FOR $, $EOD, OR $DECK RECORD
0000   110
0000   111 ;++
0000   112 ; RM$INPUT_SCAN
0000   113 ;
0000   114 ;  rm$input_scan routine to check the current (non-terminal) record for
0000   115 ;  matching the current sys$input end-of-data scan string.  this will be
0000   116 ;  either a single '$' or some user-defined string.  if the record matches
0000   117 ;  a user-defined string, cause this record to be skipped and return a
0000   118 ;  single rms$_eof error (i.e., do not latch), allowing further reads to
0000   119 ;  access subsequent records.
0000   120 ;
0000   121 ;  if matching a single '$', check if the pio$v_eod flag is on, specifying
0000   122 ;  that only a match of '$eod' is to be scanned for.  if so, then proceed as
0000   123 ;  for a match of a user-defined string above if the record contains $eod.
0000   124 ;
0000   125 ;  if not matching $eod, try for a match on $deck, and if so perform appropriate
0000   126 ;  processing.  if the record is not $deck, (i.e., it is some other record
0000   127 ;  beginning with a '$'), return rms$_eof  error and do not skip this record
0000   128 ;  so that subsequent $gets or $finds by the user will also encounter this
0000   129 ;  eof record.
0000   130 ;
0000   131 ;  if the record matches none of the above cases, simply return.  the record
0000   132 ;  will be processed normally, that is, it will be gotten or found for the user.
0000   133 ;
0000   134 ; Input Parameters:
0000   135 ;
0000   136 ;       r11     impure area addr
0000   137 ;       r10     ifab addr
0000   138 ;       r9      irab addr
0000   139 ;       r8      rab addr
0000   140 ;       r7      addr of end of buffer+1
0000   141 ;       r6      size of record in bytes
0000   142 ;       r1      addr of record in the buffer
0000   143 ;
0000   144 ; Implicit Inputs:
0000   145 ;
0000   146 ;       the contents of the various structures.
0000   147 ;       pio$gt_endstr   the current end-of-data scan string
0000   148 ;       pio$v_eod       $eod flag
0000   149 ;       note: if sys$input is from a disk file and records are allowed to
0000   150 ;             cross block boundaries, irb$b_mbc must be greater than 0,
0000   151 ;             otherwise the processing of a record crossing a block boundary
0000   152 ;             will cause rms to loop.  it is assumed that rm$connect1 has
0000   153 ;             forced mbc > 0.
0000   154 ;
0000   155 ;  (note: it has already been determined externally to this routine that
0000   156 ;   this is the sys$input stream, indirectly accessed, on a non-terminal device.
0000   157 ;
0000   158 ; outputs:
0000   159 ;
0000   160 ;       r2-r5,ap        destroyed
0000   161 ;       r0              status code
0000   162 ;
0000   163 ; Implicit Outputs:
0000   164 ;
0000   165 ;       irb$l_curbdb    may be cleared
```

```
0000   166 ;       irb$v_find      set if record is to be skipped ($eod, user-defined
0000   167 ;                       eod, or $deck)
0000   168 ;       irb$v_ppf_eof   set if $eod or user-defined eod.  after record is
0000   169 ;                       skipped, flags return of rms$_eof status.  in this
0000   170 ;                       case return status from rm$input_scan is rms$_suc.
0000   171 ;       irb$v_ppf_skip  set if record is $deck.  causes next record to be
0000   172 ;                       processed after $deck record is skipped.
0000   173 ;       irb$v_ppf_fndsv saves original state of irb$v_find when irb$v_ppf_skip
0000   174 ;                       is set
0000   175 ;
0000   176 ; completion code:
0000   177 ;
0000   178 ;       standard rms (suc or eof), except that a special code of hex 10000 will be
0000   179 ;        returned in r0 to indicate that record to be matched was not entirely
0000   180 ;        contained within the buffer.  in this case the current bdb has been
0000   181 ;        released and the buffer need merely be refilled.
0000   182 ;
0000   183 ; Side Effects:
0000   184 ;
0000   185 ;       matching the eod string resets the eod string to a single '$'.
0000   186 ;       $deck may set this to something else.
0000   187 ;       invalid syntax on '$' records will cause a hard eof, which will cause
0000   188 ;       the cli to process the offending record.
0000   189 ;
0000   190 ;--
0000   191
```

```
                          0000   193 RM$INPUT_SCAN::
                          0000   194
                          0000   195 ;
                          0000   196 ;  Assume entire record is contained in the buffer.
                          0000   197 ;
                 51   DD  0000   198          PUSHL   R1                      ; save the current record offset
                          0002   199
                          0002   200 ;++
                          0002   201 ;
                          0002   202 ;  now try to match against end of data string
                          0002   203 ;
                          0002   204 ;--
                          0002   205
          53  00000000'9F   9E  0002   206          MOVAB   @#PIO$GT_ENDSTR,R3      ; get addr of end of data string
                    52   83   9A  0009   207          MOVZBL  (R3)+,R2                ; get string length
                    52   56   B1  000C   208          CMPW    R6,R2                   ; is record at least this long?
                         06   1F  000F   209          BLSSU   SUCXIT                  ; no = no match
                63   61   52   29  0011   210          CMPC3   R2,(R1),(R3)            ; eod string match?
                         06   13  0015   211          BEQL    EOD_MATCH               ; branch if yes
                              0017   212
                              0017   213 ;
                              0017   214 ; no - simply return
                              0017   215 ;
                              0017   216
                              0017   217 SUCXIT: RMSSUC
                 02   BA  001A   218 SCNXIT: POPR    #^M<R1>                 ; restore record address
                      05  001C   219 SCNRSB: RSB
```

```
                          001D   221
                          001D   222 ;++
                          001D   223 ;
                          001D   224 ;  have matched the current end-of-data string
                          001D   225 ;
                          001D   226 ;  if matching a single '$', must check for $eod and $deck, else have
                          001D   227 ;  matched a user-defined (via $deck) string and must skip it, giving
                          001D   228 ;  a single eof error.
                          001D   229 ;
                          001D   230 ;--
                          001D   231
                          001D   232 EOD_MATCH:
    2401 8F   00000000'9F  B1   001D   233         CMPW    @#PIO$GT_ENDSTR,#1+<^A/$/@8>    ; are we matching single '$'?
                       29  12   0026   234         BNEQ    END_OF_DATA                     ; branch if not
              50   56   01  C3   0028   235         SUBL3   #1,R6,R0                        ; set remaining byte count
                          002C   236         DCL_SCAN        <EOD>,EQUAL=END_OF_DATA1 ; scan for 'eod'
                          0034   237
                          0034   238 ;
                          0034   239 ; record not $eod
                          0034   240 ;
                          0034   241
      DB 00000000'9F   01  E0   0034   242         BBS     #PIO$V_EOD,@#PIO$GW_STATUS,SUCXIT ; branch if only matching '$eod'
                          003C   243         DCL_SCAN        <DECK>,EQUAL=GOT_DECK   ; scan for 'deck'
                          0045   244
                          0045   245 ;++
                          0045   246 ;
                          0045   247 ;  have found a '$' record that is neither $eod nor $deck.
                          0045   248 ;
                          0045   249 ;  return rms$_eof and reset the nrp to find this record again.
                          0045   250 ;
                          0045   251 ;--
                          0045   252
                          0045   253 SETEOF:: RMSERR  EOF
    40 A9   48 A9   7D    004A   254         MOVQ    IRB$L_RP_VBN(R9),IRB$L_NRP_VBN(R9)
                    C9    11    004F   255         BRB     SCNXIT
```

```
                     0051   257
                     0051   258 ;++
                     0051   259 ;
                     0051   260 ;  have encounter user-defined end-of-data string
                     0051   261 ;
                     0051   262 ;  reset end-of-data scan string to match a single '$'
                     0051   263 ;
                     0051   264 ;--
                     0051   265
                     0051   266 END_OF_DATA:
00000000'9F  2401 8F B0   0051   267          MOVW    #1+<^A/$/@8>,@#PIO$GT_ENDSTR
                  0B 11   005A   268          BRB     EOD1
                     005C   269
                     005C   270 ;++
                     005C   271 ;
                     005C   272 ;  have encounter $eod
                     005C   273 ;
                     005C   274 ;  clear irb$v_ppf_eod so that any '$' record will give eof error, and
                     005C   275 ;  set irb$v_ppf_eof to cause eof to be returned after record has been skipped.
                     005C   276 ;
                     005C   277 ;--
                     005C   278
                     005C   279 END_OF_DATA1:
00000000'9F    02 8A 005C   280          BICB2   #1@PIO$V_EOD,@#PIO$GW_STATUS; clear eod flag
                  50 D5 0063   281          TSTL    R0                      ; any other tokens seen?
                  DE 12 0065   282          BNEQ    SETEOF                  ; branch if yes (error)
                     0067   283 EOD1:    SSB     #IRB$V_PPF_EOF,(R9)     ; cause eof to be returned
                     006B   284 SET_FIND:
                     006B   285          SSB     #IRB$V_FIND,(R9)        ; set find bit to skip record
                  A6 11 006F   286 SUC_BR: BRB     SUCXIT
```

```
                         0071   288
                         0071   289 ;++
                         0071   290 ;
                         0071   291 ;  have found $deck
                         0071   292 ;
                         0071   293 ;  scan for /dollars qualifier
                         0071   294 ;
                         0071   295 ;--
                         0071   296
                         0071   297 GOT_DECK:
                         0071   298         DCL_SCAN  </DOLL>,EQUAL=GOT_DOLLARS     ; scan for '/doll'
                   C8    007B   299         BNEQ    SETEOF                          ; branch if something other than
                         007D   300
                         007D   301 ;
                         007D   302 ; '/doll' seen (error)
                         007D   303 ;++
                         007D   304 ;
                         007D   305 ;  saw $deck, either with no qualifier or with /dollars and either a null or
                         007D   306 ;  no argument.
                         007D   307 ;
                         007D   308 ;  in any case, set pio$v_eod to indicate '$eod' is the locical end-of-data string
                         007D   309 ;
                         007D   310 ;--
                         007D   311 ;
                         007D   312
          00000000'9F    02     88  007D   313 SETEOD: BISB2   #1@PIO$V_EOD,@#PIO$GW_STATUS
                         0084   314 SET_SKIP:
                         0084   315         SSB     #IRB$V_PPF_SKIP,(R9)    ; set flag to skip $deck record
                         0088   316
                         0088   317 ;
                         0088   318 ; and read the next
                         0088   319 ;
                         0088   320
                DF 69    29     E1  0088   321         BBC     #IRB$V_FIND,(R9),SET_FIND; branch if doing $get
                DF 69    30     E3  008C   322         BBCS    #IRB$V_PPF_FNDSV,(R9),SUC_BR; save find bit and branch
```

```
                     0090   324
                     0090   325 ;++
                     0090   326 ;
                     0090   327 ;  have found $deck /dollars
                     0090   328 ;
                     0090   329 ;  scan for end-of-data string value indicator ('=' or ':')
                     0090   330 ;
                     0090   331 ;--
                     0090   332
                     0090   333 GOT_DOLLARS:
                     0090   334         DCL_SCAN        <:>,EQUAL=GOT_ARG       ; scan for ':'
                     0096   335
E5   13              0096   336         BEQL    SETEOD                          ; branch if nothing else in record
                     0098   337         DCL_SCAN        <=>,EQUAL=GOT_ARG       ; scan for '='
                     009E   338
A5   11              009E   339         BRB     SETEOF                          ; bad syntax - give eof error
                     00A0   340
                     00A0   341 ;++
                     00A0   342 ;
                     00A0   343 ;  have found $deck /dollars :
                     00A0   344 ;
                     00A0   345 ;  scan for end-of-data string value
                     00A0   346 ;
                     00A0   347 ;--
                     00A0   348
                     00A0   349 GOT_ARG:
                     00A0   350         DCL_SCAN        <'>,EQUAL=GOT_QUOTE     ; scan for quoted string
D5   13              00A6   351         BEQL    SETEOD                          ; branch if nothing else
                     00A8   352
                     00A8   353 ;++
                     00A8   354 ;
                     00A8   355 ;  have an unquoted, non-null end-of-data string value described by r0,r1
                     00A8   356 ;
                     00A8   357 ;  copy characters to end of data string up to first blank, tab or "!",
                     00A8   358 ;  converting them to upper case.
                     00A8   359 ;
                     00A8   360 ;--
                     00A8   361
      52   01   D0   00A8   362         MOVL    #1,R2                           ; flag unquoted string value
           02   11   00AB   363         BRB     UNQUOTED
                     00AD   364
                     00AD   365 ;++
                     00AD   366 ;
                     00AD   367 ;  have found $deck /dollars : "
                     00AD   368 ;
                     00AD   369 ;  scan for closing quote, moving characters to end of data string.
                     00AD   370 ;  process such that successive double quotes cause a single double quote
                     00AD   371 ;  to be entered into the end of data string.
                     00AD   372 ;
                     00AD   373 ;--
                     00AD   374
                     00AD   375 GOT_QUOTE:
      52   D4        00AD   376         CLRL    R2                              ; flag quoted string
                     00AF   377 UNQUOTED:
      55   D4        00AF   378         CLRL    R5                              ; build string count here
54    00000000'9F   9E   00B1   379         MOVAB   @#PIO$GT_ENDSTR,R4      ; addr of eod string length
      53   01 A4    9E   00B8   380         MOVAB   1(R4),R3                        ; addr of eod string text
```

```
                03   11   00BC    381                BRB     20$                     ; go process characters
             83 81   90   00BE    382 10$:           MOVB    (R1)+,(R3)+             ; copy char to eod string
                50   D7   00C1    383 20$:           DECL    R0                      ; any more characters?
                4C   19   00C3    384                BLSS    60$                     ; branch if not
             10 52   E8   00C5    385                BLBS    R2,45$                  ; branch if unquoted string
             22 61   91   00C8    386                CMPB    (R1),#^A/"/             ; matching quote?
                32   13   00CB    387                BEQL    50$                     ; branch if yes
          ED 55   0F   F3   00CD    388 30$:           AOBLEQ  #EODSTR_MAXLEN,R5,10$   ; count char. and branch if ok
                          00D1    389
                          00D1    390 ;++
                          00D1    391 ;
                          00D1    392 ;  exceeded max character count.  reset eod match string to single '$'.
                          00D1    393 ;
                          00D1    394 ;--
                          00D1    395
          01 A4   24   90   00D1    396 40$:           MOVB    #^A/$/,1(R4)            ; restore match string
                FF6D   31   00D5    397                BRW     SETEOF                  ; go give error
                          00D8    398
                          00D8    399 ;++
                          00D8    400 ;
                          00D8    401 ;  unquoted string
                          00D8    402 ;
                          00D8    403 ;  move character to e-o-d string unless it's blank, tab, or "!"
                          00D8    404 ;  and convert to upper case.
                          00D8    405 ;
                          00D8    406 ;--
                          00D8    407
             20   61   91   00D8    408 45$:           CMPB    (R1),#^A/ /             ; space?
                2D   13   00DB    409                BEQL    55$                     ; branch if yes
             09   61   91   00DD    410                CMPB    (R1),#TAB               ; tab?
                28   13   00E0    411                BEQL    55$                     ; branch if yes
             21   61   91   00E2    412                CMPB    (R1),#^A/!/             ; "!"?
                23   13   00E5    413                BEQL    55$                     ; branch if yes
          61 8F   61   91   00E7    414                CMPB    (R1),#LOWERCASE_A       ; lower case char?
                E0   1F   00EB    415                BLSSU   30$                     ; branch if not
          7A 8F   61   91   00ED    416                CMPB    (R1),#LOWERCASE_Z       ; well, is it?
                DA   1A   00F1    417                BGTRU   30$                     ; branch if not
          02 55   0F   F3   00F3    418                AOBLEQ  #EODSTR_MAXLEN,R5,48$   ; count char. & branch if ok
                D8   11   00F7    419                BRB     40$                     ; go process eod length error
       83   81   20   83   00F9    420 48$:           SUBB3   #LOWERCASE_A-^A/A/,(R1)+,(R3)+; convert to upper case
                C2   11   00FD    421                BRB     20$                     ; go get next char.
```

```
                     00FF    423
                     00FF    424 ;++
                     00FF    425 ;
                     00FF    426 ;  found a double quote character while processing quoted string
                     00FF    427 ;
                     00FF    428 ;  check next char. for double quote and include only one if found
                     00FF    429 ;
                     00FF    430 ;--
                     00FF    431
     50    D7        00FF    432 50$:    DECL    R0                      ; any more characters?
     0E    19        0101    433         BLSS    60$                     ; branch if not
     51    D6        0103    434         INCL    R1                      ; point to next char
  22 61    91        0105    435         CMPB    (R1),#^A/"/             ; is it another " ?
     C3    13        0108    436         BEQL    30$                     ; branch if yes
                     010A    437
                     010A    438 ;++
                     010A    439 ;
                     010A    440 ;  have completed string value, but remaining string is non-null.
                     010A    441 ;
                     010A    442 ;  give an error if anything other than blanks, tabs, or comment.
                     010A    443 ;
                     010A    444 ;--
                     010A    445
     50    D6        010A    446 55$:    INCL    R0                      ; restore character count
   0077    30        010C    447         BSBW    BLNK_SKIP               ; skip blanks and tabs
     C0    12        010F    448         BNEQ    40$                     ; branch if other than comment
                     0111    449
                     0111    450 ;++
                     0111    451 ;
                     0111    452 ;  end of data string set up o.k.
                     0111    453 ;
                     0111    454 ;  store length and go skip record.
                     0111    455 ;
                     0111    456 ;--
                     0111    457
     55    95        0111    458 60$:    TSTB    R5                      ; any chars processed?
     03    12        0113    459         BNEQ    70$
   FF65    31        0115    460         BRW     SETEOD                  ; branch if none
  64 55    90        0118    461 70$:    MOVB    R5,(R4)                 ; store count
   FF66    31        011B    462         BRW     SET_SKIP                ; go skip record
                     011E    463
```

```
                     011E   465          .SBTTL  DCL_SCAN SUBROUTINE
                     011E   466
                     011E   467 ;++
                     011E   468 ;
                     011E   469 ;  dcl_scan subroutine to scan for next token and compare it to one
                     011E   470 ;  being searched for.  case is not significant for the compare.
                     011E   471 ;  any initial blanks or tabs are skipped over.
                     011E   472 ;
                     011E   473 ;  if the strings match, the return is made to the address specified in the
                     011E   474 ;  "equal" input argument, otherwise return is made in line.
                     011E   475 ;  note that the dcl_scan macro is used to set up the in-line argument list
                     011E   476 ;
                     011E   477 ;  in the case of strings other than length 1, any characters following
                     011E   478 ;  the matched characters and before the next terminator are considered
                     011E   479 ;  to be part of the token and are also skipped in setting the remaining
                     011E   480 ;  string descriptor, as are any trailing blanks or tabs.
                     011E   481 ;
                     011E   482 ;  inputs:
                     011E   483 ;
                     011E   484 ;       r0              remaining string length
                     011E   485 ;       r1              remaining string start address
                     011E   486 ;       (sp)            counted, upper-case string to match
                     011E   487 ;       (sp)+count      branch byte offset for equal compare
                     011E   488 ;
                     011E   489 ;  outputs:
                     011E   490 ;
                     011E   491 ;       r0              length of remaining string (past token and possible
                     011E   492 ;                       trailing blanks if matched)
                     011E   493 ;       r1              address of remaining string
                     011E   494 ;       r2-r5,ap        destroyed
                     011E   495 ;
                     011E   496 ;  notes:
                     011E   497 ;
                     011E   498 ;       1. if no match, r0 & r1 will be updated to point past any initial
                     011E   499 ;          spaces and or tabs
                     011E   500 ;       2. r0 will be set to 0 on return if no string or only a comment remains
                     011E   501 ;       3. z-bit will be set based on r0
                     011E   502 ;
                     011E   503 ;--
                     011E   504
09 2F 3A 3D 21 20    011E   505 TRMLST: .ASCII  \ !=:/\<TAB>
           00000006  0124   506         TLSTSZ=.-TRMLST
                     0124   507
                     0124   508 DCL_SCAN:
          55   6E   D0  0124   509         MOVL    (SP),R5                 ; get addr of counted ascii
                     0127   510
                     0127   511 ;
                     0127   512 ;  match string
                     0127   513 ;
                     0127   514
          54   85   9A  0127   515         MOVZBL  (R5)+,R4                ; get length of string
                     012A   516
                     012A   517 ;
                     012A   518 ;  (r5 now points to string)
                     012A   519 ;
                     012A   520
          6E   54   C0  012A   521         ADDL2   R4,(SP)                 ; bump return address ...
```

```
                 6E   D6  012D    522          INCL    (SP)                    ; ... to point past ascii string
                          012F    523
                          012F    524 ;++
                          012F    525 ;
                          012F    526 ;  skip initial spaces and tabs
                          012F    527 ;
                          012F    528 ;--
                          012F    529
                 55   10  012F    530          BSBB    BLNK_SKIP               ; skip tabs and blanks
                 4E   13  0131    531          BEQL    NULL_STRING             ; branch if nothing left
                          0133    532
                          0133    533 ;++
                          0133    534 ;
                          0133    535 ;  the string desribed by r0, r1 is of non-zero length and does not begin
                          0133    536 ;  with tab, space, or "!"
                          0133    537 ;
                          0133    538 ;  see if it matches the scan string (described by r4 & r5)
                          0133    539 ;
                          0133    540 ;--
                          0133    541
              52   50   7D  0133    542          MOVQ    R0,R2                   ; save remaining len and addr
              50   54   C2  0136    543          SUBL2   R4,R0                   ; at least match count long?
                   43   19  0139    544          BLSS    NOMATCH                 ; branch if not
                   54   DD  013B    545          PUSHL   R4                      ; save match string count
              5C   81   90  013D    546 10$:     MOVB    (R1)+,AP                ; get next byte
          61  8F   5C   91  0140    547          CMPB    AP,#LOWERCASE_A         ; lower case?
                   09   1F  0144    548          BLSSU   20$                     ; branch if not
          7A  8F   5C   91  0146    549          CMPB    AP,#LOWERCASE_Z         ; well, is it?
                   03   1A  014A    550          BGTRU   20$                     ; branch if not
              5C   20   82  014C    551          SUBB2   #LOWERCASE_A - <^A/A/>,AP; convert to upper case
              85   5C   91  014F    552 20$:     CMPB    AP,(R5)+                ; match?
                   28   12  0152    553          BNEQ    UNEQUAL                 ; branch if not
                 E6 54   F5  0154    554          SOBGTR  R4,10$                  ; loop
```

```
                                0157   556
                                0157   557 ;++
                                0157   558 ;
                                0157   559 ;  strings are equal
                                0157   560 ;
                                0157   561 ;  if match count is not = 1, scan to end of token and then to start of next token
                                0157   562 ;  (end of token is indicated by space, tab, !, /, =, or :)
                                0157   563 ;
                                0157   564 ;  in any case, take the "equal=" exit
                                0157   565 ;
                                0157   566 ;++
                                0157   567 ;--
                                0157   568
                          8E    D7  0157   569          DECL    (SP)+                           ; was match count = 1?
                          18    13  0159   570          BEQL    60$                             ; branch if yes
                    52    50    7D  015B   571          MOVQ    R0,R2                           ; save remaining descriptor
                          07    11  015E   572          BRB     40$
                B9 AF     06    83  0160   573 30$:     LOCC    (R3)+,#TLSTSZ,TRMLST            ; is character a delimiter?
                          03    12  0165   574          BNEQ    50$                             ; branch if yes
                       F6 52    F4  0167   575 40$:     SOBGEQ  R2,30$                          ; loop if more characters
                          52    D6  016A   576 50$:     INCL    R2                              ; don't count terminator
                          53    D7  016C   577          DECL    R3                              ; or point past it
                    50    52    7D  016E   578          MOVQ    R2,R0                           ; descriptor to right regs
                          13    10  0171   579          BSBB    BLNK_SKIP                       ; go skip tabs and blanks
                 52    00 BE    98  0173   580 60$:     CVTBL   @(SP)+,R2                       ; pick up "equal" branch offset
                    6E    52    C0  0177   581          ADDL2   R2,(SP)                         ; add in offset to return pc
                          07    11  017A   582          BRB     SCAN_XIT
                                0157   583
                                0157   584 ;++
                                0157   585 ;
                                0157   586 ;  the input string didn't contain the match string
                                0157   587 ;
                                0157   588 ;  leave r0,r1 describing any remaining string and take in-line (non-equal) return
                                0157   589 ;
                                0157   590 ;--
                                0157   591
                                017C   592 UNEQUAL:
                          BE    D5  017C   593          TSTL    (SP)+                           ; pop saved match count
                                017E   594 NOMATCH:
                50    52    7D  017E   595          MOVQ    R2,R0                           ; restore save descriptor
                                0181   596 NULL_STRING:
                          6E    D6  0181   597          INCL    (SP)                            ; skip past "equal" return offset
                                0183   598 SCAN_XIT:
                          50    D5  0183   599          TSTL    R0                              ; set z bit according to r0
                                05  0185   600          RSB
```

```
                     0186   602
                     0186   603 ;++
                     0186   604 ;
                     0186   605 ;  blnk_skip subroutine to skip past blanks and tabs, up to possible comment
                     0186   606 ;  or end of input string
                     0186   607 ;
                     0186   608 ;  inputs:
                     0186   609 ;
                     0186   610 ;       r0      input string length
                     0186   611 ;       r1      input string address
                     0186   612 ;
                     0186   613 ;  outputs:
                     0186   614 ;
                     0186   615 ;       r0      remaining string length after blanks and tabs skipped
                     0186   616 ;               (if only a comment left, r0 will be set to zero)
                     0186   617 ;       r1      remaining string address
                     0186   618 ;       z-bit   set if no more input (other than comment), else clear
                     0186   619 ;
                     0186   620 ;--
                     0186   621
                     0186   622 BLNK_SKIP:
 61   50   20   3B   0186   623 10$:    SKPC    #^A/ /,R0,(R1)          ; skip spaces
           12   13   018A   624         BEQL    30$                     ; branch if nothing but spaces
      61   09   91   018C   625         CMPB    #TAB,(R1)               ; is char tab?
           06   12   018F   626         BNEQ    20$                     ; branch if not (done)
           50   D7   0191   627         DECL    R0                      ; yes - decrement count
           51   D6   0193   628         INCL    R1                      ; skip tab
           EF   11   0195   629         BRB     10$                     ; and continue skipping
      61   21   91   0197   630 20$:    CMPB    #^A/!/,(R1)             ; do we have a comment?
           02   12   019A   631         BNEQ    30$                     ; branch if not
           50   D4   019C   632         CLRL    R0                      ; yes - say end of input
                05   019E   633 30$:    RSB                             ; return with z-bit set if no
                     019F   634
                     019F   635 ;
                     019F   636 ; more input
                     019F   637 ;
                     019F   638
                     019F   639
                     019F   640         .END
```

```
$$.PSECT_EP                     = 00000000
$$RMSTEST                       = 0000001A
$$RMS_PBUGCHK                   = 00000010
$$RMS_TBUGCHK                   = 00000008
$$RMS_UMODE                     = 00000004
BLNK_SKIP                         00000186 R      01
DCL_SCAN                          00000124 R      01
END_OF_DATA                       00000051 R      01
END_OF_DATA1                      0000005C R      01
EODT                              00000067 R      01
EODSTR_MAXLEN                   = 0000000F
EOD_MATCH                         0000001D R      01
GOT_ARG                           000000A0 R      01
GOT_DECK                          00000071 R      01
GOT_DOLLARS                       00000090 R      01
GOT_QUOTE                         000000AD R      01
IRB$L_NRP_VBN                   = 00000040
IRB$L_RP_VBN                    = 00000048
IRB$V_FIND                      = 00000029
IRB$V_PPF_EOF                   = 0000002E
IRB$V_PPF_FNDSV                 = 00000030
IRB$V_PPF_SKIP                  = 0000002F
LOWERCASE_A                     = 00000061
LOWERCASE_Z                     = 0000007A
NOMATCH                           0000017E R      01
NULL_STRING                       00000181 R      01
PIO$GT_ENDSTR                     ********   X    01
PIO$GW_STATUS                     ********   X    01
PIO$V_EOD                       = 00000001
RM$INPUT_SCAN                     00000000 RG     01
RMS$_EOF                        = 0001827A
SCAN_XIT                          00000183 R      01
SCNRSB                            0000001C R      01
SCNXIT                            0000001A R      01
SETEOD                            0000007D R      01
SETEOF                            00000045 R      01
SET_FIND                          0000006B R      01
SET_SKIP                          00000084 R      01
SUCXIT                            00000017 R      01
SUC_BR                            0000006F R      01
TAB                             = 00000009
TLSTSZ                          = 00000006
TRMLST                            0000011E R      01
UNEQUAL                           0000017C R      01
UNQUOTED                          000000AF R      01
```

+----------------+
! Psect synopsis !
+----------------+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
RM$RMS1	0000019F (415.)	01 (1.)	PIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC BYTE
ABS	00000000 (0.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.09	00:00:00.93
Command processing	132	00:00:00.70	00:00:05.80
Pass 1	271	00:00:07.54	00:00:22.25
Symbol table sort	0	00:00:00.93	00:00:02.03
Pass 2	122	00:00:02.08	00:00:08.15
Symbol table output	7	00:00:00.06	00:00:00.55
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	571	00:00:11.42	00:00:39.74

The working set limit was 1200 pages.
42967 bytes (84 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 747 non-local and 28 local symbols.
640 source lines were read in Pass 1, producing 14 object records in Pass 2.
20 pages of virtual memory were used to define 19 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[RMS.OBJ]RMS.MLB;1	10
_255DUA28:[SYS.OBJ]LIB.MLB;1	0
_255DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	14

832 GETS were required to define 14 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:RM1INPSCN/OBJ=OBJ$:RM1INPSCN MSRC$:RM1INPSCN/UPDATE=(ENH$:RM1INPSCN)+EXECML$/LIB+LIB$:RMS/LIB
```

RM1CONN
LIS

RM1GET
LIS

RM1UNPSCN
LIS

RM1DISCON
LIS

RM1GETINT
LIS

RM1CREATE
LIS

RM1JOURNL
LIS